Affichage et manipulation interactive de formules mathématiques dans les documents structurés

Hanane Naciri, Laurence Rideau

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Affichage et manipulation interactive de formules mathématiques dans les documents structurés

Hanane Naciri — Laurence Rideau

N° 4140

Mars 2001

THÈME 2

ISSN 0249-6399 ISRN INRIA/RR--4140--FR

INRIA
SOPHIA ANTIPOLIS

Affichage et manipulation interactive de formules mathématiques dans les documents structurés

Hanane Naciri [*] , Laurence Rideau [†]

Thème 2 — Génie logiciel
et calcul symbolique
Projet Lemme

Rapport de recherche n° 4140 — Mars 2001 — 31 pages

Résumé : Afficher des formules mathématiques et interagir avec ces formules sont des atouts primordiaux pour les outils informatiques dédiés aux mathématiques. Dans ce rapport, nous faisons un bilan des outils existants puis nous décrivons FIGUE, moteur d'affichage interactif incrémental et bidimensionnel, développé à l'INRIA, pour obtenir une bibliothèque dédiée au développement d'éditeurs de documents structurés et d'interfaces graphiques. Enfin nous montrons un exemple d'utilisation de FIGUE, dans le cadre du développement de preuves mathématiques sur ordinateur.

Mots-clés : formules mathématiques, outils de formatage, MathML, édition structurée, document numérique, interaction homme machine

[*] Hanane.Naciri@sophia.inria.fr
[†] Laurence.Rideau@sophia.inria.fr

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, B.P. 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : 04 92 38 77 77 - International : +33 4 92 38 77 77 — Fax : 04 92 38 77 65 - International : +33 4 92 38 77 65

Display and Interactive Manipulation of Mathematic Formulas in Structured Documents

Abstract: Tools dedicated to mathematics need to display formulas and to interact with them. In this report, we present a summary of existing tools, then we describe FIGUE, an incremental two dimensional layout engine , developed at INRIA, to get a specialized toolbox for building customized editors and graphical user interfaces. Finally we give an example of interface using FIGUE to develop mathematical proofs on computer.

Key-words: mathematics formulas, formatting tools, MathML, structured edition, electronic document, man-machine interface

Table des matières

1 Introduction

La production de documents scientifiques et en particulier le développement d'environnements graphiques pour les mathématiques nécessitent la possibilité d'exprimer et de manipuler une grande diversité d'objets (texte simple, formules mathématiques, schémas, ...). Des problèmes particuliers se posent pour les formules mathématiques qui ont une structure complexe et bidimensionnelle (fractions, matrices, intégrales,...).

Nous distinguons deux familles de systèmes de production de documents scientifiques :

- les systèmes d'édition de documents en mode non-interactif (*batch-oriented processors*), comme par exemple le système TEX [KNU 90] qui permet un formatage de grande qualité ;
- les éditeurs WYSIWYG (*What You See Is What You Get*), comme par exemple AMAYA [VAT 00] (éditeur WYSIWYG de pages Web permettant une édition structurée de formules mathématiques).

Ces derniers éditeurs offrent un environnement graphique qui aide l'utilisateur à visualiser son document pendant toute la phase de création. Ils permettent de manipuler dynamiquement des objets graphiques contrairement aux outils de la première famille qui nécessitent la transcription du document à l'aide d'un langage qui sera interprété ultérieurement pour l'affichage.

Parmi les éditeurs WYSIWYG qui intègrent les formules mathématiques citons Edimath [QUI 83, QUI 84], Publisher [MCC 87], FrameMaker [COR 91], Grif [QUI 86] et Word [ARB 93]. D'autre part, plusieurs éditeurs mathématiques ont été développés pour permettre l'édition des expressions mathématiques usuelles : formule éditée et formatée peut ensuite être copiée en tant qu'image dans d'autres applications. Parmi les éditeurs connus de ce type, citons MacEqn [OF 85], MathWriter [COO 86], Expressionist [KAL 89], MathType [SCI 87] et EquationBuilder [TAL 92].

Les systèmes de production de document, genre Grif [QUI 86] ou INForm [EGM 89], ajoutent à l'approche WYSIWYG les avantages de l'édition structurée. Contrairement aux autres systèmes déjà cités, ces systèmes séparent la représentation du document de son contenu logique, ce qui permet une interaction avec les objets du documents. L'éditeur Euromath [CHL 98], basé sur Grif, offre un environnement de travail pour les mathématiciens utilisant un modèle de données uniforme. L'auteur du document doit seulement s'occuper du contenu, l'affichage et la structure étant manipulés par le système. Cependant, ces systèmes de production de documents manipulent uniquement des expressions mathématiques éditées par l'utilisateur et non pas des formules dont la structure n'est pas connue d'avance, comme dans le cas où de telles expressions seraient générées par un système de calcul symbolique extérieur. Des environnements graphiques pour des systèmes de calcul symbolique

doivent alors manipuler à la fois des expressions saisies par l'utilisateur et celles générées par le système de calcul.

Des expérimentations d'interface d'outils mathématiques, ont tout d'abord été menées dans le domaine du calcul formel : interface spécifique à Maple [ROS 95], ou plus généralement interface pour le calcul symbolique [KAJ 93], dont les idées ont ensuite été largement reprises [SOI 95, KAJ 98] : les systèmes de calcul formel (Maple, Mathematica, ...) sont aujourd'hui dotés d'interfaces très conviviales et puissantes.

La plupart de ces travaux (en particulier les travaux de Kajler [KAJ 93]) s'appuient sur l'édition structurée qui est apparue dans les années 80. Elle a été appliquée aux outils d'aide à la programmation (comme le Cornell Synthesizer [TEI 81] ou Centaur [BOR 87]). Basés sur des descriptions syntaxiques et sémantiques d'un langage donné, ces outils avaient comme objectifs la génération d'environnements de programmation spécifiques au langage décrit. Ces environnements dotés d'interfaces graphiques puissantes incluaient un système d'édition structurée, des interpréteurs et divers autres outils sémantiques comme la transformation de programmes ou la traduction. Plus tard, le savoir-faire en matière d'édition structurée spécifique à ces outils d'aide à la programmation a été appliqué à la construction d'environnements pour les systèmes de preuves [THE 92, BER 94, BER 99].

En général, tout outil pour les mathématiques doit assurer un affichage performant de formules mathématiques (bidimensionnel, incrémental, facilement personnalisable). Le développement d'outils de formatage génériques et portables participe à améliorer la qualité d'affichage et d'interaction dans les environnements WYSIWYG pour les mathématiques. De tels outils de formatage doivent non seulement résoudre le problème de l'affichage d'une diversité d'objets comme les formules mathématiques, mais également fournir un moyen simple et naturel pour interagir avec ces objets. Ces outils doivent aussi pouvoir permettre de partager les données mathématiques avec d'autres applications et de les diffuser sur Internet en adoptant par exemple le format standard pour les expressions mathématiques MathML (dialecte XML pour les formules mathématiques)

C'est dans cette optique que nous avons développé le système FIGUE [FIG], qui est un module de formatage et d'affichage interactif. La suite de ce document en décrira les principales composantes : Les caractéristiques de FIGUE pour manipuler des objets structurés, en particulier les formules mathématiques, seront expliquées en détail; ensuite, nous présenterons le standard MathML et comment il est intégré à FIGUE; enfin, nous montrerons le rôle de FIGUE pour le développement d'environnements graphiques pour des systèmes de calcul symbolique.

2 Affichage et manipulation d'objets structurés

2.1 Présentation de la bibliothèque FIGUE

FIGUE est un module de formatage et d'affichage interactif et portable utilisé dans le développement d'outils WYSIWYG comme le développement d'éditeurs de documents structurés (des systèmes de production de document). FIGUE permet de présenter les objets structurés et en particulier les formules mathématiques de façon conviviale (avec une bonne qualité d'affichage) et offre des interactions variées à la souris comme la sélection de sous-expressions afin de pouvoir les manipuler dynamiquement (évaluation, simplification, modification, génération de code, etc). D'autre part, FIGUE est un module indépendant, offrant la possibilité de présenter graphiquement des objets structurés, édités par les utilisateurs en suivant une syntaxe suffisamment naturelle et non ambiguë, ou produits par des calculs de systèmes symboliques; il permet aussi de les manipuler dynamiquement et les modifier. FIGUE est donc, en particulier, un excellent candidat pour le développement d'interfaces pour les mathématiques. Il est aujourd'hui utilisé dans l'équipe LEMME [1] de l'INRIA Sophia-Antipolis pour le développement d'outils interactifs et d'éditeurs d'objets structurés comme les programmes, les formules mathématiques ou les preuves sur ordinateur.

Au départ, une version préliminaire de FIGUE, intégrée au système CENTAUR, a été utilisée pour le développement de CAS/PI [KAJ 93], une interface utilisateur générique pour les systèmes de calcul formel. Actuellement, FIGUE (écrit totalement en JAVA) est utilisé pour le développement du système PCOQ, une interface graphique conçue pour le système de développement de preuves COQ [HUE 97] et aussi dans le projet Smarttools [SMA] pour le développement d'outils interactifs génériques.

Dans cette section, nous présentons en détails les caractéristiques de FIGUE pour manipuler et afficher des objets structurés, en particulier les formules mathématiques: les structures de boîtes, les constructeurs graphiques, les algorithmes de formatage et d'affichage, les possibilités d'incrémentalité, la gestion de sélection à la souris et les mécanismes d'interaction.

2.2 Structures de boîtes

FIGUE se base sur la notion de boîtes [KNU 90] pour représenter tous les objets structurés du document. Il manipule un arbre de boîtes produit par une description d'un langage abstrait, appelé PPML (*Pretty Printing Meta Language*) [JAC 94]. Ce formalisme PPML était proposé par le système CENTAUR pour exprimer l'affichage des données (objets) structurées.

1. Logiciels et Mathématiques

Une description PPML est une suite de règles où chaque règle associe à une structure logique de données sa structure de boîtes ou de représentation décrivant le formatage correspondant. PPML transforme les objets représentés sous forme d'arbre de syntaxe abstraite (suivant un formalisme) en un arbre de boîtes FIGUE correspondant à l'affichage : pour chaque noeud de l'arbre de syntaxe PPML cherche et applique la règle de transformation correspondante. La figure 1 montre un exemple de transformation d'un arbre de syntaxe abstraite représentant une formule mathématique en un arbre de boîtes FIGUE.

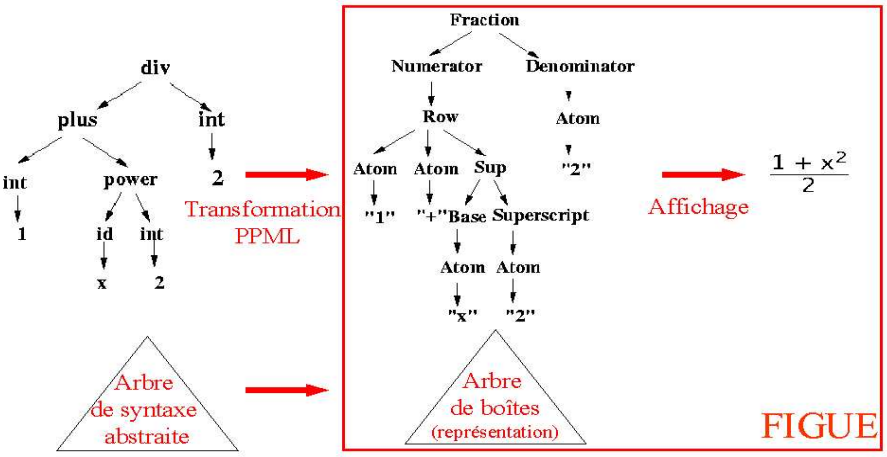


FIG. 1 – *Transformation d'un arbre de syntaxe abstraite représentant une formule mathématique en un arbre de boîtes* FIGUE*, à l'aide de règles* PPML.

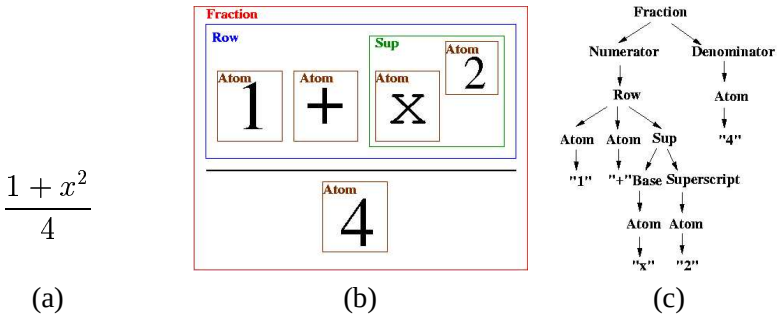
Prenons un exemple simple d'une règle PPML décrivant comment l'opérateur binaire d'addition sera affiché :

$$\text{plus}(*x, *y) \quad \rightarrow \quad [\text{<Row>} *x \text{ "+" } *y]$$

On associe à un arbre de syntaxe abstraite dont la racine est un "plus" ayant deux fils $*x$ et $*y$, une boîte composée d'un constructeur graphique *Row* (voir section suivante) et trois éléments $*x$, $+$ et $*y$. En partie droite de la règle, $*x$ et $*y$ sont des appels récursifs de l'affichage sur les fils de l'opérateur plus. Ils seront séparés par la chaîne "+" qui est un atome (une feuille de l'arbre de boîtes). Le constructeur *Row* permet de formater ses fils en

mode horizontal et applique des règles de coupure si la boîte englobante dépasse la largeur de la zone d'affichage.

Les objets du document sont représentés sous forme d'arbre de boîtes, où chaque nœud est une boîte (constructeur graphique) englobant toutes ses boîtes filles et les feuilles sont des atomes (unités basiques comme les chaînes de caractères). La figure 2 montre un exemple simple de représentation en boîtes d'une formule mathématique.



(a) (b) (c)

FIG. 2 – *(a) Formule mathématique. (b) Ensemble de boîtes correspondantes. (c) Représentation sous forme d'arbre de boîtes.*

2.3 Constructeurs graphiques

FIGUE offre, par défaut, quelques constructeurs graphiques. Chaque constructeur prend une liste de boîtes ou des atomes en argument et les formate selon un algorithme de formatage spécifique. Les constructeurs graphiques de base de formatage en FIGUE sont : *Atom*, *Horizontal*, *Vertical* et *Paragraphe*. *Atom* est un constructeur de base qui produit des feuilles de l'arbre de boîtes (atomes qui n'ont pas de fils). *Horizontal* formate en mode horizontal la liste de ses fils. Il prend en paramètre l'espace entre deux éléments. L'alignement se fait au niveau du point de sortie d'un élément et du point d'entrée de l'élément suivant (voir figure 3). *Vertical* dispose ses fils en mode vertical. Il prend en paramètre l'espace consécutif entre deux lignes et l'indentation (i.e. un décalage horizontal de tous les éléments par rapport au premier). *Paragraphe* met ses éléments en mode horizontal tant qu'il reste assez de marge à droite de la page. Sinon il retourne à la ligne et dispose à nouveau ses éléments horizontalement (*Paragraphe* correspond au constructeur *Row* décrit dans la section précédente)

En utilisant ces constructeurs de base, nous arrivons seulement à gérer une édition linéaire des documents (comme des programmes). Ce résultat n'est pas suffisant pour mani-

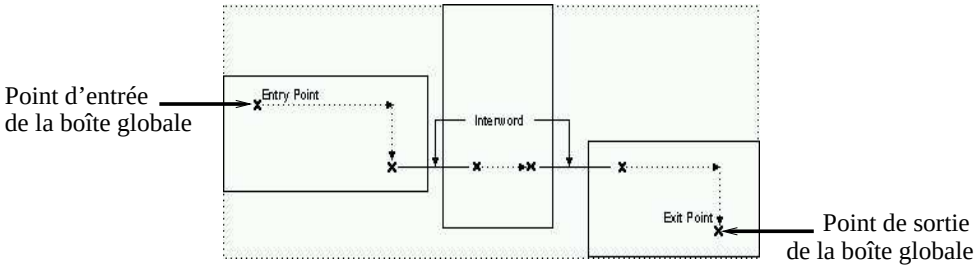


FIG. 3 – *Le constructeur Horizontal dispose ses fils horizontalement. Pour son alignement avec les autres boîtes, il a le même point d'entrée que son premier fils et le point de sortie de son dernier fils.*

puler des formules mathématiques qui sont naturellement bidimensionnelles (intégrale, matrice, racine carrée,. . .). Pour cela, nous avons introduit dans FIGUE de nouveaux constructeurs pour les formules mathématiques (*Racine*, *Puissance*, *Matrice*,. . .). Ces constructeurs mathématiques disposent leurs fils selon la formule mathématique correspondante et au besoin dessinent les symboles nécessaires. Par exemple, le constructeur *Racine* dessine le signe racine autour de sa première boîte fille en tenant compte de sa taille (voir figure 4) et dispose son deuxième fils de façon à avoir une racine n-ième.

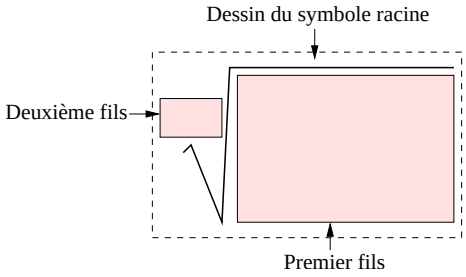


FIG. 4 – *Disposition des boîtes et le dessin de symboles pour le constructeur racine (Le formatage de la racine n-ième).*

2.4 Formatage et affi chage bidimensionnels

Le formatage et l'affichage au niveau de FIGUE sont bidimensionnels, par opposition aux méthodes employées dans certaines interfaces qui obligent à linéariser les notations.

Ils correspondent aux notations employées pour les documents manuscrits ou typographiés. FIGUE permet la disposition de boîtes dans un espace à deux dimensions (voir figure 5) en respectant la spécificité d'étirement de certains constructeurs en fonction de leur contenu (racine, intégrale, fraction).

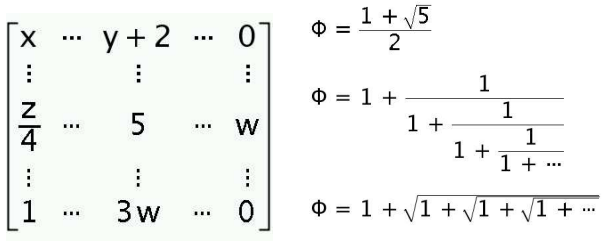


FIG. 5 – *Exemples de formatage bidimensionnel d'objets. (a) Disposition des éléments d'une matrice. (b) Affichage de formules mathématiques de structures différentes.*

Le formatage en FIGUE manipule directement l'arbre de boîtes. Cette structure d'arbre mémorise les relations de dépendances (la hiérarchie d'inclusion) entre les boîtes (objets graphiques) à afficher. L'algorithme de formatage parcourt cet arbre en mettant à jour les propriétés (ou attributs) graphiques des boîtes : origine, taille, alignement (voir figure 3). Les propriétés graphiques d'une boîte sont déterminées par la nature de son constructeur graphique et en fonction des propriétés de ses boîtes filles.

L'algorithme de formatage prend aussi en compte les paramètres de la zone d'affichage, i.e la largeur de la page (pour que les boîtes soient visibles) et les différents paramètres liés au contexte graphique de chaque boîte (les multiples polices de caractères utilisées).

Chaque constructeur a son propre algorithme de formatage, déterminant comment il disposera ses fils par rapport à son origine, ce qui ensuite déterminera l'alignement de sa boîte englobante avec l'ensemble (avec ses points d'entrée et de sortie - voir figure 3). Un soin particulier est accordé à l'efficacité des algorithmes de formatage des constructeurs mathématiques. Par exemple, la disposition correcte des éléments d'une matrice requiert un algorithme en plusieurs itérations [2].

Une fois le formatage effectué, chaque boîte FIGUE est capable de s'afficher elle-même dans un contexte graphique (polices de caractères, couleur, arrière-plan, coordonnées). L'algorithme d'affichage parcourt simplement l'arbre de boîtes pour afficher les boîtes en fonc-

2. La position d'un terme de la matrice n'est connu qu'après avoir déterminé, par une recherche plus globale, la largeur de chaque colonne de la matrice et la hauteur de chaque ligne (voir fi gure 5a).

tion de leur contexte graphique et éventuellement dessiner de nouveaux symboles ou des caractères typographiques (comme par exemple le symbole racine - voir figure 3).

2.5 Incrémentalité

En mode interactif, nous devons minimiser le coût de reformatage dû à la mise à jour ou à la sélection d'une ou plusieurs boîtes. Chaque modification d'une boîte (contexte, taille, origine,. . .) doit être propagée dans l'ensemble de l'arbre de façon optimale afin de mettre à jour seulement les boîtes affectées. Le formatage dans FIGUE peut être effectué de façon incrémentale, i.e. quand une boîte est déplacée ou modifiée, il ne recalcule que les propriétés modifiées (positionnement, origine, taille, rang des fils) des boîtes concernées. Ce formatage incrémental conduit à un affichage incrémental des objets du document où seules les zones modifiées ou touchées par la modification sont réaffichées.

Dans la version actuelle de FIGUE, l'algorithme de formatage incrémental est relativement simple. En mettant à jour la structure de l'arbre de boîtes, les modifications sont propagées vers la racine de père en père. Cet algorithme utilise les propriétés particulières de chaque boîte. Par exemple, dans le cas d'une boîte qui dispose ses fils horizontalement, nous ne reformatons qu'à partir du premier fils modifié (les autres fils seront poussés vers la droite, voir figure 6). Dans FIGUE, il est possible de spécialiser l'algorithme de reformatage pour chaque boîte.

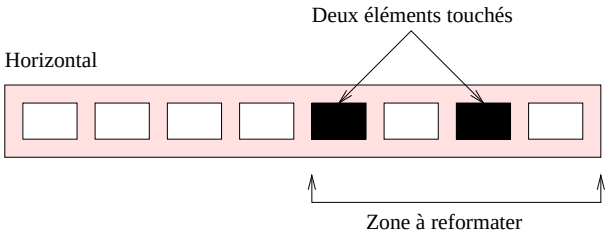


FIG. 6 – *Reformatage incrémental de la boîte Horizontal.*

L'incrémentalité dépend de la sémantique (ou la nature) de chaque constructeur graphique. Pour chaque constructeur, il faut se poser la question suivante : quels sont les éléments à reformater si on modifie une boîte fille de ce constructeur. Nous étudions avec soin l'algorithme incrémental de chaque constructeur FIGUE et en particulier les constructeurs mathématiques où la solution n'est pas évidente à obtenir. Par exemple, dans le cas du constructeur matrice, il est difficile de décider après changement d'un élément de la matrice quelles sont les éléments à reformater (voir figure 5a).

2.6 Interaction et sélection

La plupart des interfaces modernes et interactives offrent la possibilité de sélectionner à la souris les objets affichés afin de les manipuler dynamiquement (en utilisant le copier-coller entre autres).

FIGUE offre un puissant mécanisme d'interaction permettant de manipuler dynamiquement et d'interagir à la souris avec les objets affichés (calcul, simplification de formules mathématiques, génération de code, . . .). Nous distinguons deux niveaux d'interaction (voir figure 7). Le premier niveau d'interaction se situe entre l'utilisateur et les objets du document affiché par l'interface graphique et le deuxième niveau d'interaction se situe entre l'interface graphique et des systèmes extérieurs effectuant des traitements et des calculs (des systèmes de calcul symbolique ou autres).

FIG. 7 – *Deux niveaux d'interaction existent en* FIGUE. *L'utilisateur peut sélectionner des objets du document. La structure de ces objets est utilisée pour communiquer ensuite une requête à un système de calcul.*

FIGUE permet à l'utilisateur de sélectionner à la souris les objets affichés grâce à son mécanisme de sélection directement lié à la structure de l'arbre de boîtes. Il traduit automatiquement les coordonnées de la souris sur l'écran en un emplacement dans la structure d'arbre de boîtes et retrouve ainsi le sous-arbre correspondant aux objets sélectionnés (voir figure 8). Une fois que la boîte (objet graphique) est sélectionnée par l'utilisateur, il est possible d'obtenir le sous-arbre de la structure syntaxique sous-jacente correspondant à cette boîte[3]. Le lien entre la structure de représentation des objets du document et leur contenu syntaxique est maintenu par le système, ce qui offre un mécanisme puissant d'interaction structurée.

3. On détermine la règle PPML qui est à l'origine de l'affi chage de cette boîte. Ensuite, le sous-arbre sélectionné correspond alors au schéma syntaxique de la partie gauche de la règle PPML

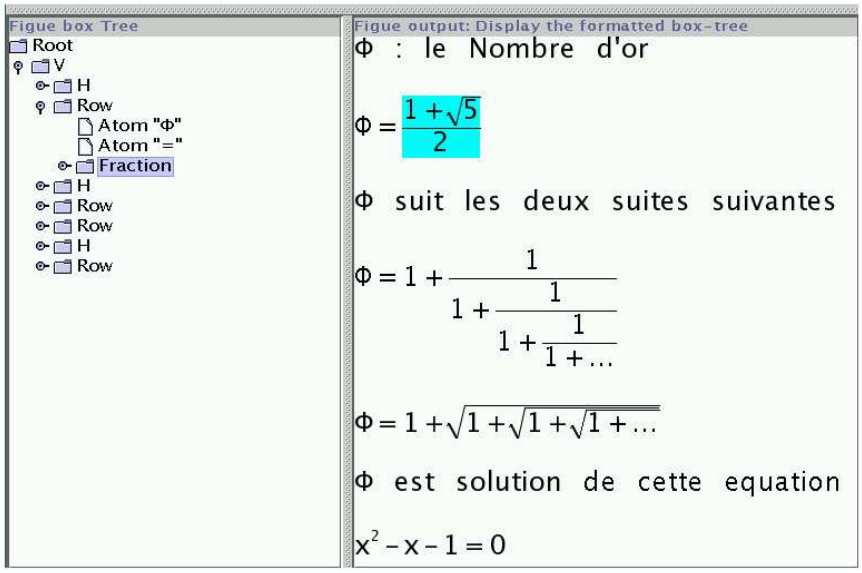


FIG. 8 – *Sélection des objets affichés par l'utilisateur dans la structure du document (partie droite), et calcul de leurs emplacements dans la structure d'arbre de boîtes par* FIGUE *(partie gauche).*

La sélection peut être simple (un seul élément) ou multiple (plusieurs éléments sélectionnés à la fois). A chaque sélection, une action peut être appliquée, comme dessiner la plus petite boîte englobante contenant l'objet sélectionné. Par exemple si on a l'expression mathématique $(a * c) + b$ et qu'on sélectionne le symbole $*$, la sous-expression $a * c$ sera automatiquement sélectionnée.

Dans le cas d'une sélection multiple de boîtes, il n'y a pas forcément de sous-arbre directement associé. Il est néanmoins possible de formuler des heuristiques qui permettent une sélection multiple structurée, comme par exemple en déterminant la plus petite sous-expression contenant tous les objets sélectionnés. Si on reprend l'expression mathématique $(a*c)+b$ et qu'on sélectionne cette fois a et c, la sous-expression $a*c$ sera sélectionnée, car elle représente la plus petite sous-expression englobant les objets sélectionnés. Par contre, si on sélectionne c et b alors l'expression $(a * c) + b$ sera sélectionnée entièrement.

Malheureusement, dans le cas où l'on veut sélectionner la diagonale d'une matrice, cette méthode sélectionne toute la matrice (la plus petite structure contenant tous les éléments sélectionnés). Toutefois, il est possible d'affiner la méthode de sélection et de construire

une sous-liste *virtuelle* des éléments de la diagonale sur laquelle l'interface pourra travailler comme si cette sous-liste existait vraiment dans l'objet structuré sous-jacent.

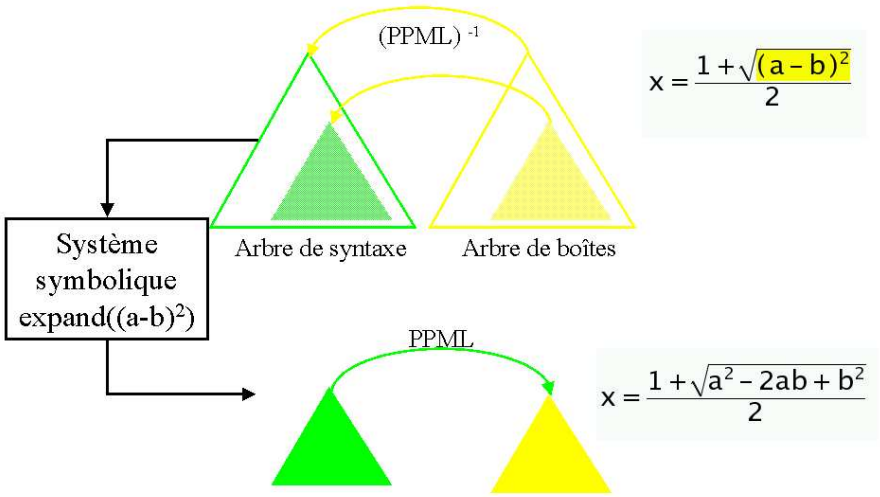


FIG. 9 – *Le mécanisme d'interaction permet d'obtenir la syntaxe des objets affichés afin de la communiquer au système de calcul symbolique et ensuite de réafficher le résultat du calcul.*

Il est également possible d'effectuer des opérations plus *intelligente* qu'un copier-coller, comme par exemple développer une expression mathématique. La figure 9 illustre ce mécanisme d'interaction plus complexe. L'utilisateur veut développer la sous-expression mathématique $(a-b)^2$ en utilisant l'interface graphique. Premièrement, l'utilisateur sélectionne la sous-expression à développer. Ensuite, l'interface récupère la structure syntaxique de cette sous-expression et la communique à un système de calcul symbolique via un protocole de communication défini (ce qui demande de traduire la structure de la sous-expression selon ce protocole). Après traitement, ce système renvoie le résultat du développement sous forme d'arbre de syntaxe. L'interface remplacera l'ancienne sous-expression par ce résultat. Pour cela, l'interface utilisera les règles de transformation PPML spécifiques pour obtenir le sous-

arbre de boîtes correspondant. Les mécanismes de formatage et d'affichage incrémentaux sont alors exploités (voir section 2.5).

L'interaction et la sélection sont des fonctionnalités très importantes et très utiles dans les interfaces homme-machine. Le système PCOQ [PCO] utilise FIGUE et exploite ce mécanisme puissant de sélection pour effectuer des preuves par sélection et des transformations de formules (voir section 4.3).

2.7 Formules mathématiques

Parmi les fonctionnalités dans des systèmes dédiés aux mathématiques (section 1), notre développement de FIGUE tente de résoudre les problèmes liés à l'affichage et la manipulation de formules mathématiques. Le traitement des objets mathématiques soulève plusieurs problèmes [SOI 95] [ARS 97], comme la complexité et la diversité des règles typographiques [ATT 97], la gestion des grandes formules, l'efficacité et l'incrémentalité des algorithmes de formatage, la sélection de sous-expressions et l'ambiguïté des notations mathématiques.

Parmi les problèmes rencontrés en FIGUE liés à la complexité des règles typographiques, citons l'exemple du dessin des délimiteurs (tels que $($, $\int$,$\{$) qui sont des symboles mathématiques de taille variable délimitant des expressions mathématiques plus ou moins grandes. Le dessin de ces symboles doit tenir compte de leur taille et de leur contexte graphique [AND 93]. Pour les dessiner, nous nous sommes inspirés du système METAFONT en LaTeX [KNU 86] [KNU 89] et des polices *outlines* [HER 93] qui se basent sur les courbes de Bézier[4]. L'algorithme de dessin de chaque symbole calcule l'épaisseur et les paramètres des courbes représentant le contour du caractère en fonction de la taille de la police utilisée et de la hauteur du symbole. Ces symboles mathématiques de tailles variables sont gérés au cas par cas. Arriver à trouver un algorithme général pour résoudre les problèmes du dessin de ces symboles demande d'étudier en profondeur les règles typographiques. Pour l'instant, le résultat obtenu pour l'affichage de ces symboles est largement satisfaisant en attendant la sortie et l'intégration dans notre système de polices vectorielles appropriées.

Pour la gestion de grandes formules, plusieurs solutions sont envisageables :

- Affichage à échelle réduite de l'expression mathématique. Cette solution n'assure pas la visibilité de l'expression.
- Appliquer des règles de césure et tenter de visualiser l'expression dans son intégralité. Certaines formules (telle les matrices) supportent mal la césure.

4. Ces symboles peuvent être des courbes quelconques, par exemple des polynômes du troisième degré (cubiques) dont les courbes de Bézier sont un cas particulier.

– Utiliser l'élision pour n'afficher que les termes les plus significatifs de l'expression. Le choix des termes à retenir est difficile et nécessite des compétences mathématiques approfondies.
– Fragmentation de l'expression en sous-expressions de tailles plus raisonnables.

En FIGUE, il est possible d'associer des actions (coupure, réduction d'échelle, ...) aux grandes formules, comme appliquer des règles de coupures automatiques ou afficher les termes jusqu'à une profondeur fixée par l'utilisateur. Dans la suite de notre travail, nous espérons proposer des solutions intéressantes à ce problème difficile.

Un autre défi pour un outil de formatage et d'affichage est de permettre à l'utilisateur d'obtenir une qualité d'impression sur papier le plus proche possible de ce qui est affiché à l'écran. Cet aspect est en cours de développement en FIGUE, avec une génération de PostScript acceptable.

Enfin, dans le but de permettre un échange de données mathématiques entre applications et de l'intégration de nos outils sur internet, nous avons développé un module de comptabilité avec le format standard MathML (dialecte de XML dédié aux expressions mathématiques). Ce module sera détaillé dans la section suivante.

3 Format d'échange de données: MathML

3.1 Introduction

Un des intérêts majeurs des mathématiques sur ordinateur est de pouvoir partager les données mathématiques avec d'autres utilisateurs, de les échanger entre applications et de les diffuser sur Internet. Cela permet d'atteindre une population plus large d'utilisateurs potentiels et de partager des résultats (feuilles de calcul pour des systèmes de calcul formel, preuves, ...). Dans ce but, plusieurs formats et protocoles d'échanges de formules mathématiques existent avec des systèmes comme OpenMath [DAL 97], MathLink [RES 92] pour le système Mathematica, et plus récemment MathML [MAT], dédié aux mathématiques sur Internet.

MathML est proposé par le W3C comme un format standard pour les expressions mathématiques remplaçant les divers formats de données actuels. Certains de ces formats existant sont basés sur la syntaxe (LaTeX, notations utilisées dans les systèmes de calcul symbolique tels que Mathematica, Maple V, etc), mais jusqu'ici sur Internet, les formules mathématiques sont affichés en images (sans structure).

MathML est conçu pour servir de support d'échange entre logiciels scientifiques et mathématiques sans avoir pour unique objectif leur représentation graphique. MathML utilise deux types de description des formules : les éléments de présentation et les éléments de des-

cription sémantique (contenu). Les éléments de présentation sont destinés à la description bidimensionnelle des expressions mathématiques, alors que les éléments de contenu visent à donner une sémantique aux objets mathématiques (proche de la syntaxe abstraite).

Plusieurs communautés scientifiques s'intéressent au standard MathML : navigateurs Internet, systèmes de calcul, éditeurs structurés et éditeurs de textes. Les navigateurs Web essaient d'introduire dans leurs systèmes les formules mathématiques codées en MathML. Parmi les navigateurs Web qui supporte actuellement MathML, citons le navigateur et éditeur Amaya [VAT 00] du W3C, ainsi que les dernières versions de Mozilla, Techexplorer d'IBM et Internet Explorer.

Quelques systèmes de calcul (symbolique ou numérique) utilisent MathML pour décrire leurs résultats de calcul et les échanger avec des interfaces graphiques ou avec d'autres applications mathématiques. Le système Mathematica est capable d'évaluer des expressions MathML et d'afficher le résultat par son interface graphique. Il peut être utilisé comme un serveur de calcul pour MathML. Une expérience de formalisation des termes de preuves du système COQ [HUE 97] utilise MathML pour représenter les formules mathématiques [ASP 00a]. MathML est également choisi pour représenter dans un environnement graphique les algorithmes du calcul numérique [LI 00] en donnant la possibilité à l'utilisateur de définir ses propres algorithmes et structures de données.

D'autre part, quelques travaux autour des éditeurs de textes intègrent MathML. Le système Omega [PLA 00], une généralisation du système TEX, permet la génération automatique du MathML à partir des formules mathématiques écrites en langage TEX ou Latex. Amaya [VAT 00] permet l'édition structurée du MathML sur Internet. MathType [TOP 99] est un éditeur d'équations mathématiques et un outil d'affichage pour MathML. WebEq [WEB] offre une collection d'outils d'édition et d'affichage MathML incluant un éditeur visuel, un traducteur de WebTeX vers MathML et une *applet* d'affichage interactif des mathématiques sur Internet.

Dans le cadre de FIGUE, nous avons choisi dans un premier temps de traiter uniquement les éléments de présentation de MathML et nous envisageons par la suite d'étudier l'intégration des éléments de description sémantique dans notre système. Cette intégration devrait être simplifiée par la philosophie de notre système qui s'appuie sur la syntaxe abstraite des objets manipulés (très proche des éléments de type *contenu* de MathML). FIGUE permet l'affichage des objets mathématiques MathML et leur manipulation. FIGUE peut alors être utilisé comme base pour le développement d'éditeur d'objets structurés MathML, dans le genre d'Amaya. Il aura alors le rôle de passerelle entre les interfaces des applications et Internet.

3.2 Implémentation de MathML dans FIGUE

FIGUE implémente du MathML : il permet d'afficher les éléments de présentation MathML et de les manipuler. Inversement, il est possible de générer du MathML à partir des objets mathématiques affichés par FIGUE même si ces objets ont été crées indépendamment de MathML. Tout ceci fait que MathML peut être utilisé par FIGUE comme format d'importation et d'exportation de données et aussi comme format d'échange avec d'autres logiciels mathématiques.

FIGUE, comme MathML, utilise une notation structurée pour représenter les objets mathématiques. La correspondance entre les boîtes FIGUE et les éléments de présentation MathML se fait donc naturellement. Chaque élément de présentation MathML correspond à une boîte (constructeur graphique) FIGUE décrivant comment ses fils doivent être disposés et affichés. La figure 10 montre un exemple du constructeur *Fraction* qui dispose le numérateur au-dessus du dénominateur et sépare les deux par une barre de fraction : la figure 10 montre d'abord le code source XML incluant la représentation de la fraction en MathML (figure 10a), l'arbre de boîtes FIGUE correspondant (figure 10b) et son affichage par FIGUE (figure 10c).

Nous avons développé en FIGUE un module permettant l'interprétation des documents structurés XML décrivant les boîtes FIGUE et incluant des expressions MathML. Il s'agit premièrement d'effectuer l'analyse syntaxique de l'ensemble du document XML suivant une grammaire spécifiée par sa DTD (*Document Type Definition*) afin d'en extraire l'arbre DOM (*Document Object Model*) décrivant la structure de ce document XML. Ensuite, notre module traduit l'arbre DOM en un arbre de boîtes FIGUE qui pourra ensuite être affiché par FIGUE (voir figure 11). Ce module assure un lien bidirectionnel entre la structure d'arbre de boîtes FIGUE et la structure d'arbre DOM du document XML. Ayant le chemin d'un noeud dans l'un des arbres, il est possible d'obtenir le chemin correspondant dans l'autre arbre ; i.e si on sélectionne une expression affichée, il est possible d'obtenir l'emplacement de l'élément MathML correspondant dans l'arbre DOM et vice versa. Ceci permet d'obtenir le source XML associé à un élément affiché.

Nous avons testé deux approches pour implémenter ce module en FIGUE (voir figure 12) :

- traduire directement en code Java les objets mathématiques MathML en des objets structurés FIGUE (figure 12a) ;
- traduire à l'aide des règles de transformation paramétrées les objets mathématiques MathML en des objets structurés FIGUE (figure 12b).

La première implémentation de ce module permet de traduire directement l'arbre DOM du document contenant des éléments MathML en un arbre de boîtes FIGUE (voir figure 12a). Pour chaque élément MathML, nous associons le sous-arbre de boîtes FIGUE correspondant et nous assurons, par codage, le lien entre les deux structures.

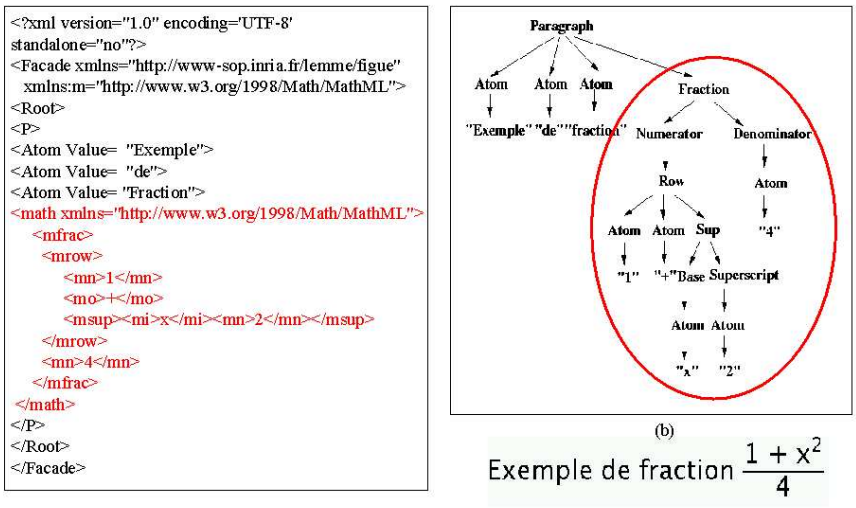


FIG. 10 – *Exemple de fraction: (a) encodage MathML, (b) arbre de boîtes* FIGUE *(c) résultat du formatage.*

La deuxième implémentation applique des règles de transformation d'un protocole nommé XPPML [5] pour traduire chaque élément MathML (ou plus généralement des éléments XML) en un élément FigueML (élément XML décrivant une boîte FIGUE) qui sera ensuite à son tour transformé en une boîte FIGUE pour être affiché (voir figure 12b). Ces règles XPPML décrivent au moyen d'un document XML les règles de conversion d'un arbre source en un autre arbre résultat. Le compilateur des règles XPPML s'appuie sur les idées du compilateur de règles PPML (voir section 2.2). Il maintient la correspondance entre l'arbre source et l'arbre résultat. La figure 13 montre quelques règles XPPML utilisées pour traduire le précédent exemple de fraction codé en MathML (voir figure 10) en un arbre de boîtes FIGUE. La première règle XPPML de cet exemple associe à l'élément MathML *mfrac* l'élément représentant la boîte de fraction en FIGUE.

5. Pour les lecteurs connaissant XML, notons que XPPML est une forme de XSLT, qui permet de transformer des fi chiers XML

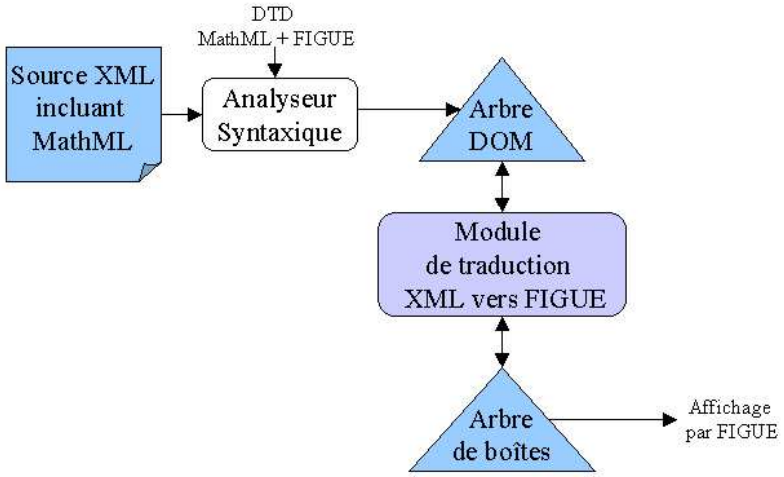


FIG. 11 – *Module d'interprétation de MathML vers* FIGUE *et vice versa. Le document contenant MathML est d'abord analysé afin d'en extraire l'arbre DOM décrivant le contenu du document. Il est ensuite possible d'effectuer la correspondance entre cet arbre et l'arbre de boîtes* FIGUE.

La deuxième implémentation est plus flexible pour l'utilisateur, malgré sa complexité apparente. Elle donne la possibilité de paramétrer la traduction des objets mathématiques MathML vers des objets FIGUE à l'aide des règles de transformation. L'utilisateur peut spécifier ses propres règles de traduction et les modifier selon ses besoins.

4 Application: développement d'interfaces graphiques

Comme nous l'avons vu précédemment, FIGUE est une composante dédiée à l'affichage interactif d'objets structurés. Il peut donc être utilisé pour le développement d'interfaces pour les mathématiques et en particulier pour des systèmes de calcul symbolique. Dans cette section, nous faisons d'abord un tour d'horizon des interfaces graphiques pour les systèmes de calcul symbolique existants. Ensuite, nous présentons notre approche générique pour le développement de telles interfaces. Enfin, nous présentons en détails l'interface graphique PCOQ qui suit cette approche générique et qui utilise FIGUE comme module d'affichage interactif.

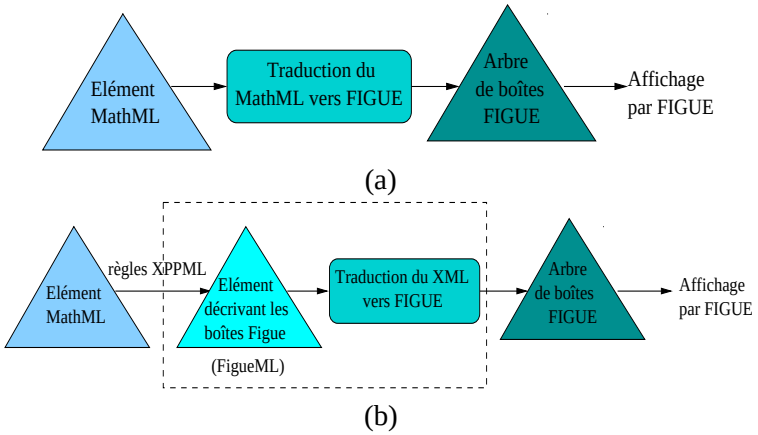


FIG. 12 – *Deux implémentations possibles pour interpréter du MathML dans* FIGUE. *(a) Schéma de traduction directe d'un élément MathML en un objet structuré* FIGUE. *(b) Schéma de traduction par utilisation de règles XPPML sur un élément MathML pour obtenir un objet structuré* FIGUE.

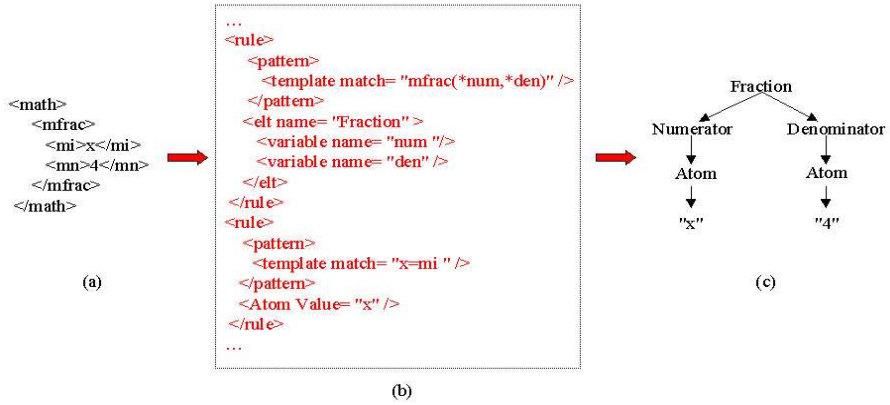


FIG. 13 – *Un exemple de traduction d'un élément MathML en un objet structuré* FIGUE *par application de règles XPPML. (a) élément MathML. (b) règles XPPML. (c) objet structuré* FIGUE.

4.1 Interfaces graphiques existantes

La plupart des systèmes de calcul symbolique offre la possibilité à l'utilisateur d'effectuer des traitements symboliques interactifs à travers une interface graphique (simplification, factorisation, preuve de théorème, ...). Ces interfaces conviviales et intuitives ont le rôle de faire abandonner le crayon et le papier aux mathématiciens, de rendre les systèmes de calcul symbolique directement accessibles aux débutants mais aussi riches de fonctionnalités pour des utilisateurs experts.

Les systèmes de calcul symbolique nécessitent donc des interfaces graphiques puissantes pour atteindre une population plus large d'utilisateurs. Dans ce but, plusieurs expérimentations d'interfaces d'outils mathématiques ont tout d'abord été faites dans le domaine du calcul formel puis pour les systèmes de preuves.

La première interface graphique réelle dans le domaine du calcul formel (interface pour le système Reduce) est MathScribe [SMI 86]. La principale caractéristique de cette interface est l'édition en deux dimensions de formules mathématiques. L'interface GI/S [YOU 87] de même type que MathScribe est une interface graphique pour le système Macsyma. Elle utilise une structure pour l'affichage qui n'est pas forcément liée à la structure de données des systèmes de calcul formel. Cela permet un mécanisme de sélection permettant par exemple la sélection des termes consécutifs d'une expression linéaire, comme la sélection de $b + c*$ dans l'expression $a - b + c * d$ même si $b + c*$ n'a pas de signification mathématique. L'interface Milo (intégrée plus tard dans FrameMaker) est basée sur la notion de document intégrant dans la même fenêtre du texte, des formules mathématiques et des traces de courbes. Milo est la première interface implémentant la manipulation directe d'expressions mathématiques. Parmi les prototypes d'interfaces graphiques génériques et distribués, CAS/PI [KAJ 92] permet à un utilisateur expert d'adapter l'interface pour un besoin spécifique et de la connecter à d'autres logiciels externes. Mathematica [BUC 93] est un système de calcul algébrique conçu sous la forme d'un noyau algébrique et d'une interface. Mathematica a apporté une nouvelle notion de cahier interactif ou "note book" qui étend le modèle de l'interface-document en permettant une navigation type hypertexte. Enfin, citons Emath [ARS 99] est une composante réutilisable et paramétrable pour l'édition de formules mathématiques qui peut être intégrée dans des interfaces de calcul formel.

Pour les systèmes de preuves, les premiers efforts pour la conception des interfaces ont commencé avec le système Nuprl [CON 86] et IPE [RIT 88] (Interactive Proof Editor). Ces deux interfaces ont introduit l'édition structurée pour maintenir et éditer des preuves. CTCOQ [BER 99], une interface utilisateur pour le système de preuves COQ [HUE 97], utilise l'environnement de programmation Centaur [BOR 87] et a introduit de nouvelles idées d'interaction, comme *proof-by-pointing* [BER 94], *drag-and-drop* [BER 97] et la gestion du script. Notons que ces outils spécifiques à l'interface CTCOQ ont ensuite été repris

dans le cas d'outils génériques comme Proof General [ASP 00b]. D'autres interfaces graphiques ont été conçues pour des systèmes de preuves existant : XIsabelle [OZO 97] une interface pour le système Isabelle [PAU 94], TkHOL [SYM 95] interface pour le système HOL [GOR 93] et le système graphique JAPE [BOR 99]. Ces systèmes dépendent de boîtes à outils graphiques orientées texte et sont moins efficaces pour la manipulation des formules à la souris. Pcoq [PCO] en cours de développement, reprend les idées de son antécédent CTCOQ en introduisant des améliorations au niveau de la manipulation des objets de preuves (voir pour plus de détails la section 4.3).

4.2 Approche générique pour le développement des interfaces graphiques

Dans notre approche générique pour le développement des interfaces graphiques pour des systèmes de calcul symbolique, l'interface est totalement séparée du noyau du calcul. Les principaux modules participant au développement de ces interfaces graphiques sont :

- un protocole d'échange de données entre l'interface graphique et le système de calcul symbolique;
- une édition et une manipulation structurées des données;
- un formatage et un affichage interactifs des objets structurés, facilement personnalisable par l'utilisateur.

La figure 14 montre notre schéma général pour le développement des interfaces graphiques utilisant FIGUE. L'interface et le système de calcul symbolique sont deux processus indépendants qui communiquent via un protocole. Ce protocole s'occupe du transfert des données, principalement des arbres, et traduit la structure des objets manipulés par l'interface en une structure de données du système de calcul symbolique et vise versa (exemple de protocole [DER 94]). L'interface graphique utilise un module d'édition structurée offrant la possibilité à l'utilisateur de saisir des commandes et un analyseur produisant des objets structurés à partir des données saisies en suivant un formalisme (définition par des types abstraits d'un langage). Les objets structurés peuvent être sauvegardés dans un fichier, communiqués au système de calcul symbolique pour effectuer des calculs ou traduit en une structure de représentation (arbre de boîtes) pour pouvoir les afficher par notre moteur d'affichage FIGUE. Le lien entre les deux structures (arbre de syntaxe abstraite et arbre de boîtes FIGUE) se fait grâce aux règles de traduction PPML (voir section 2.2). Ce lien permet un mécanisme puissant de sélection d'objets manipulés par l'interface.

L'interface graphique PCOQ, s'appuie sur ce schéma général pour le développement de son environnement de travail graphique et interactif. La section suivante décrit en détails l'architecture de PCOQ et comment elle utilise FIGUE pour l'affichage des commandes et des formules mathématiques manipulés par l'interface.

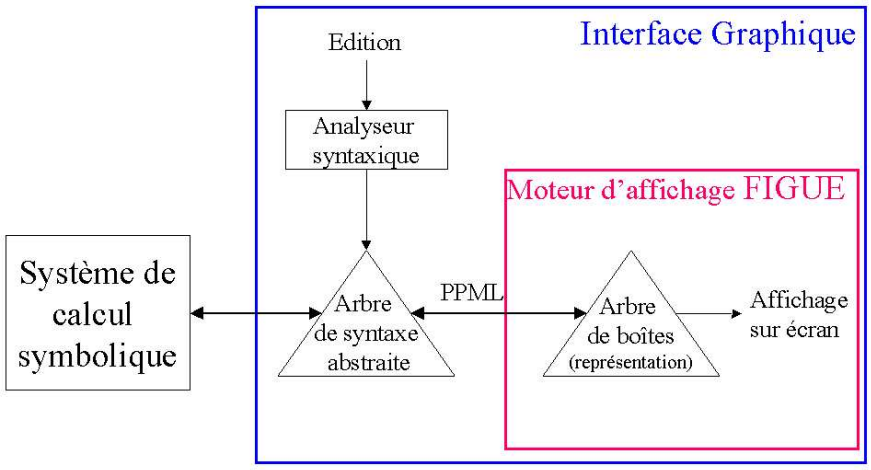


FIG. 14 – *Le schéma général des interfaces graphiques utilisant* FIGUE *pour les systèmes de calcul symboliques*

4.3 PCOQ : un exemple d'utilisation de FIGUE

PCOQ (la version JAVA de CTCOQ [BER 99] en cours de développement [PCO]) fournit un environnement de travail graphique et interactif pour le système de preuve COQ [HUE 97] suivant l'architecture décrite précédemment. Le moteur de preuve et l'interface utilisateur sont deux processus indépendants. Le but principal de PCOQ est d'aider au développement de preuves à grande échelle et de fournir une interface conviviale facilitant la création de preuves pour les utilisateurs du système de preuve COQ. Cette interface possède les caractéristiques suivantes :

- une interface graphique : présentation structurée et colorée des formules et des commandes ;
- des mécanismes d'édition et de présentation structurées : l'environnement fournit les moyens d'éditer structurellement formules et commandes ;
- la preuve par sélection : l'environnement utilise la structure des formules logiques pour aider systématiquement l'utilisateur à guider la preuve par de simples sélections à la souris [BER 94].

L'interface graphique se base sur la boîte à outils pour la manipulation des données structurées AÏOLI et le noyau d'affichage FIGUE. Les données venant de COQ sont transformées en une structure d'arbres. Ensuite, AÏOLI manipule cette structure en utilisant ses trois composantes principales, inspirées des concepts du système CENTAUR [BOR 87] :

- VTP (*Virtual Tree Processor*) : une machine abstraite permettant de spécifier et de manipuler des structures d'arbres en suivant un formalisme (définition par des types abstraits d'un langage).
- PPML (*Pretty Printing Meta Language*) : un langage permettant de décompiler un arbre de syntaxe abstraite en un arbre de boîtes. Une spécification PPML est une suite de règles où chaque règle se compose d'une partie gauche représentant un arbre de syntaxe abstraite et une partie droite décrivant le formatage correspondant [JAC 94].
- GFXOBJ : une librairie d'objets graphiques (menus, fenêtres,...) permettant le développement d'applications interactives pour les arbres VTP.

AÏOLI, à l'aide de PPML, traduit un arbre de syntaxe abstraite en un arbre de boîtes. A partir de cet arbre de boîtes, FIGUE construit l'arbre de boîtes formaté qui sera affiché par la suite.

La figure 15 montre un exemple de session PCOQ (preuve en COQ que $\sqrt{2}$ n'est pas rationnel : $\sqrt{2} \notin \mathbb{Q}$). Dans cet exemple, différentes notations sont manipulées : texte, formules mathématiques et images. L'utilisateur peut effectuer des preuves par sélection [BER 94], i.e. la sélection d'une formule logique permet la génération automatique de la commande à envoyer à COQ (en fonction du contexte, de la position de l'expression sélectionnée, ...). PCOQ offre aussi la possibilité de sélectionner et de déplacer un terme d'une formule (*drag and drop* [BER 97]) en appliquant la transformation correspondante. Si nous prenons l'exemple simple de l'équation $ax + b = 0$ et que nous déplaçons le paramètre b du côté droit, nous obtenons l'équation équivalente $ax = -b$ (si la théorie mathématique sous-jacente le permet; i.e. on est dans un anneau).

5 Conclusion

Tout au long de cet article, nous avons fait apparaître les principales fonctionnalités que tout outil pour les mathématiques devrait assurer :

- un affichage performant de formules mathématiques (bidimensionnel, incrémental, facilement personnalisable);
- une manipulation simple et naturelle des formules (interactivité);
- des moyens d'entrées des données naturels et efficaces (saisie);

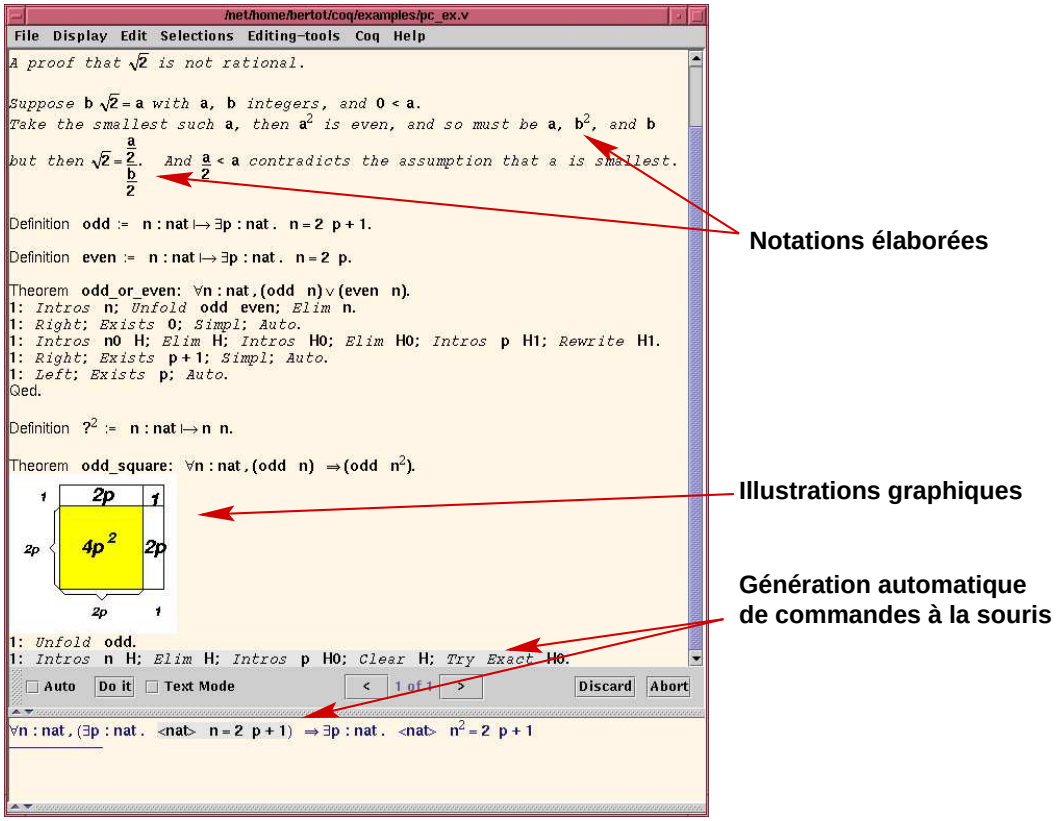


FIG. 15 – *Une session* PCOQ.

– une impression donnant un résultat sur papier le plus proche possible de ce qui est affiché sur l'écran (ou même meilleur!);
– des moyens de transférer et de communiquer des données.

Lors de nos développements, nous avons essayé de respecter ces différents besoins. Bien que nous nous soyons focalisés sur les aspects *affichage et interaction* d'une part et *transfert et communication* d'autre part, nous étudions aussi comment générer du PostScript dans le but d'obtenir une qualité d'impression sur papier acceptable. En ce qui concerne la *saisie* de formules, le moyen le plus simple que nous ayons à notre disposition est le passage par un format Ascii (saisie au clavier); pour cela il faut utiliser une syntaxe suffisamment naturelle et non ambiguë, mais malheureusement différente de ce qui est affiché (syntaxe de LaTeX par exemple). Dans un futur proche, nous voulons aussi donner la possibilité de saisir

les formules à travers des menus, en prévoyant des raccourcis clavier pour alléger la tâche des utilisateurs plus expérimentés. Des expériences ont aussi été menées dans le cadre de la reconnaissance de formules manuscrites [KOS 99, OKA 99, OHT 00] saisies grâce à une tablette graphique, mais ces travaux n'ont pas encore été intégrés à notre système.

Le but principal de notre développement de FIGUE est donc essentiellement de produire une bibliothèque de formatage et d'affichage dédiée au développement d'outils interactifs (WYSIWYG) pour les mathématiques comme le développement d'éditeur de documents scientifiques ou des interfaces graphiques pour les systèmes de calcul symbolique. Dans cette optique, notre approche ne se contente pas seulement d'offrir un affichage de qualité d'objets structurés comme les formules mathématiques mais elle offre aussi un moyen puissant et naturel pour interagir et manipuler ces objets actifs du document. Aujourd'hui FIGUE a dépassé le stade de prototype et est utilisé, en particulier, comme brique de base de PCOQ, à qui il fournit un affichage et une interaction très performants, adaptables facilement aux besoins de l'utilisateur.

D'autre part, FIGUE intègre un module permettant l'interprétation et la manipulation des documents structurés XML incluant du MathML, ce qui permet l'échange de formules avec d'autres applications. Pour le futur, nous nous fixons comme objectif d'utiliser ce module de FIGUE pour développer un éditeur d'objets structurés MathML pour le Web, dans le genre de AMAYA. L'avantage de notre approche est d'associer les objets à leur sens sémantique. Contrairement à l'approche de la plupart des éditeurs qui manipulent les objets en tant que texte, notre méthode offre la possibilité de faire des traitements sur les objets affichés hors du contexte du document, comme par exemple, la possibilité de simplifier une formule mathématique d'un document Web.

Enfin, à partir du travail réalisé en PCOQ sur la mise en place d'explications des preuves en langue naturelle [COS 96], une autre direction de recherche est envisageable pour nous. Aujourd'hui, il existe des explications en langue française et anglaise. Dans l'avenir, nous envisageons d'expérimenter l'implémentation des explications en arabe, ce qui permettrait de tester et de pousser dans ses limites FIGUE pour écrire de droite à gauche et même mélanger l'affichage droite-gauche (texte arabe) et l'affichage gauche-droite (mathématiques).

Références

[AND 93] ANDRE J., VATTON I., Contextual typesetting of mathematical symbols taking care of optical scaling, Technical Report, RR-1972, 1993, Inria.

[ARB 93] ARBOR A., ARBORTEXT M.,CORPORATION I., Word User's Guide, 1993.

[ARS 97] ARSAC O., Interfaces homme machine pour le calcul formel , PhD thesis, Université de Nice Sophia Antipolis, Juillet 1997.

[ARS 99] ARSAC O., DALMAS S.,GAËTANO M., The Design of a Customizable Component to Display and Edit Formulas , DOOLEY S., *Proceedings of the 1999 International Symposium on Symbolic and Algebraic Computation (ISSAC-99)*, New York, jul 1999, ACM Press, p. 283–290.

[ASP 00a] ASPERTI A., PADOVANI L., COEN C. S.,SCHENA I., Formal Mathematics in MathML , *MathML International Conference*, 2000, extended abstract.

[ASP 00b] ASPINALL A., Proof General: A Generic Tool for Proof Development. , 1785 L., *Tools and Algorithms for the Construction and Analysis of Systems, Proc TACAS 2000*, 2000.

[ATT 97] ATTAR P., Prendre en compte les notations mathématiques dans les documents électroniques , *Document numérique*, vol. 1, n. 2, 1997, p. 147–159.

[BER 94] BERTOT Y., KAHN G.,THERY L., Proof by Pointing , *Lecture Notes in Computer Science*, vol. 789, 1994, p. 141.

[BER 97] BERTOT Y., Direct manipulation of Algebraic Formulae in Interactive Proof Systems , *Workshop on User-Interfaces for Theorem Provers*, Sophia Antipolis, sep 1997.

[BER 99] BERTOT Y., The CtCoq System: Design and Architecture , *Formal aspects of Computing*, vol. 11, 1999, p. 225-243.

[BOR 87] BORRAS P., CLEMENT D., DESPEYROUX T., INCERPI J., KAHN G., LANG B.,PASCUAL V., CENTAUR: The system , Research Report n. 777, Decembre 1987, INRIA, Rocquencourt, France.

[BOR 99] BORNAT R.,SUFRIN B., Animating Formal Proof at the Surface: The Jape Proof Calculator , *The Computer Journal*, vol. 42, n. 3, 1999, p. 177–192.

[BUC 93] BUCHBERGER B., *Mathematica:* A System for Doing Mathematics by Computer? , Technical Report n. 93-50, 1993, RISC-Linz, Johannes Kepler University, Linz, Austria.

[CHL 98] CHLEBÍKOVÁ J., The Euromath System - The Structured Editor for Mathematicians , *Proceedings of the Tenth European TeX Conference*, , 1998, p. 82–93.

[CON 86] CONSTABLE R. L., ALLEN S., H. BROMELY, CLEVELAND W., *Implementing Mathematics with the Nuprl Development System*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1986.

[COO 86] COOKE J. R.,SOBEL E. T., *MathWriter: mathematical typesetting with the Macintosh*, Cooke Publications, Ithaca, 1986.

[COR 91] CORPORATION F., JOSE S.,USA C., FrameMaker Reference Manual , 1991.

[COS 96] COSCOY Y., A Natural Language Explanation for Formal Proofs , RETORÉ C., *Proceedings of Int. Conf. on Logical Aspects of Computational Liguistics (LACL), Nancy*, vol. 1328, Springer-Verlag LNCS/LNAI, September 1996.

[DAL 97] DALMAS S., GAËTANO M.,S.WATT, An OpenMath 1.0 implementation , KÜCHLIN W. W., *ISSAC '97. Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation, July 21–23, 1997, Maui, Hawaii*, New York, NY 10036, USA, 1997, ACM Press, p. 241–248.

[DER 94] DERY A.,RIDEAU L., Distributed programming environments: an example of a message protocol , Technical Report n. RT-0165, 1994, Inria.

[EGM 89] VAN EGMOND S., HEEMAN F.,VAN VLIET J., INFORM: an interactive syntax-directed formulae editor , *The Journal of Systems and Software*, vol. 9, n. 3, 1989, p. 169–182.

[FIG] Figue , `http://www-sop.inria.fr/croap/figue`.

[GOR 93] GORDON M. J. C.,MELHAM T. F., *Introduction to HOL: A theorem proving environment for higher order logic*, Cambridge University Press, 1993.

[HER 93] HERSCH. R. D., *Visual and Technical Aspects of Type*, Cambridge University Press, 1993.

[HUE 97] HUET G., KAHN G.,PAULIN-MOHRING C., The Coq Proof Assistant : A Tutorial : Version 6.1 , Technical Report n. RT-0204, 1997, Inria.

[JAC 94] JACOBS I.,RIDEAU-GALLOT L., The PPML Manual , Technical report, August 1994, Inria.

[KAJ 92] KAJLER N., CAS/PI: a portable and extensible interface for computer algebra systems , WANG P. S., *Proceedings of ISSAC '92. International Symposium on Symbolic and Algebraic Computation*, New York, NY 10036, USA, 1992, ACM Press, p. 376–386.

[KAJ 93] KAJLER N., User interfaces for Symbolic Computation: a Case Study , *Proceedings of the 6th Annual Symposium on User Interface Software and Technology*, New York, NY, USA, Novembre 1993, ACM Press, p. 1–10.

[KAJ 98] KAJLER N.,SOIFFER N., A Survey of User Interfaces for Computer Algebra Systems , *Journal of Symbolic Computation*, vol. 25, n. 2, 1998, p. 127–160.

[KAL 89] KALTOFEN E.,WATT S., *Computers and Mathematics*, SpringerVerlag, 1989.

[KNU 86] KNUTH D., *Computers and Typesetting*, Addison-Wesley, Reading, 1986.

[KNU 89] KNUTH D., Typesetting *Concrete Mathematics* , *TUGboat*, vol. 10, n. 1, 1989, p. 31–36.

[KNU 90] KNUTH D., *The TeX-Book (revised)*, Addison Wesley, 1990.

[KOS 99] KOSMALA A., LAVIROTTE S., POTTIER L.,RIGOLL G., On-Line Handwritten Formula Recognition using Hidden Markov Models and Context Dependent Graph Grammars , *5th International Conference on Document Analysis and Recognition*, Bangalore, India, sep 1999.

[LI 00] LI J.,LETT G. S., Using MathML to Describe Numerical Computations , *MathML International Conference*, 2000, extended abstract.

[MAT] MathML , `http://www.w3.org/Math/`.

[MCC 87] MCCARTHY E., HOLLAND C.,LEHMAN J., The Publisher User Manual , 1987.

[OF 85] OF F., SOFTWARE S., VENABLE, MACEQN D., A COMPUTER,P ROCHESTER, New York: Software for Recognition Technologies , 1985.

[OHT 00] OHTAKE N., FUKUDA R.,SUZUKI M., Optical Recognition and Braille Transcription of Mathematical Documents , *7th International Conference on Computers Helping People with Special Needs ICCHP*, Karlsruhe, 2000.

[OKA 99] OKAMURA H., KANAHORI T., CONG W., FUKUDA R., TAMARI F.,SUZUKI M., Handwriting Interface for Computer Algebra Systems , *Fourth Asian Technology Conference in Mathematics*, Guangzhou, 1999, p. 291-300.

[OZO 97] OZOLS M. A., CANT A.,EASTAUGHFFE K. A., XIsabelle: A System Description , MCCUNE W., *Proceedings of the 14th International Conference on Automated deduction*, vol. 1249 *LNAI*, Berlin, jul 1997, Springer, p. 400–403.

[PAU 94] PAULSON L. C., Isabelle: A Generic Theorem Prover , *Lecture Notes in Computer Science*, vol. 828, 1994, p. xvii + 321.

[PCO] Pcoq , `http://www-sop.inria.fr/croap/pcoq`.

[PLA 00] PLAICE O.,HARALAMBOUS Y., Generating MathML and Other ...ML from Omega , *MathML International Conference*, 2000, extended abstract.

[QUI 83] QUINT V., An Interactive System for Mathematical Text Processing , *Technology and Science of Informatics*, vol. 3, n. 2, 1983, p. 169-179.

[QUI 84] QUINT V., Interactive Editing of Mathematics , *Proceedings of the First International Conference on Text Processing Systems*, Boole Press Limited, 1984, p. 55–68.

[QUI 86] QUINT V., VATTON I.,BEDOR H., GRIF: An interactive environment for TEX , DÉSARMÉNIEN J., *TEX for Scientific Documentation. Second European Confe-*

rence. Strasbourg, France, Lecture Notes in Computer Science, Berlin, Germany / Heidelberg, Germany / London, UK / etc., 1986, Springer-Verlag, p. 145–158.

[RES 92] RESEARCH W., MathLink reference guide. , 1992, Wolfram Research.

[RIT 88] RITCHIE B., The Design and Implementation of an Interactive Proof Editor , PhD thesis, 1988.

[ROS 95] ROSIER P., Développement d'un environnement de programmation Centaur pour Maple , Rapport de stage de DEA Calcul et Déduction, Université de Nice - Sophia Antipolis, 1995, UNSA.

[SCI 87] SCIENCE D., MathType User Manual , 1987.

[SMA] SmarTools , `http://www-sop.inria.fr/oasis/SmartTools/`.

[SMI 86] SMITH C. J.,SOIFFER N., MathScribe: A User INterface for Computer Algebra Systems , *Proceedings of the 1986 Symposium on Symbolic and Algebriaic Computation*, jul 1986, p. 7–13.

[SOI 95] SOIFFER N., The Design of a User Interface for Computer Algebra Systems , Technical Report n. CSD-91-626, 1995, University of California, Berkeley.

[SYM 95] SYME D., A New Interface for HOL — Ideas, Issues and Implementation , *Lecture Notes in Computer Science*, vol. 971, 1995, p. 324.

[TAL 92] TALBOT T., EquationBuilder User Manual , 1992.

[TEI 81] TEITELBAUM T.,REPS T., The Cornell Program Synthesizer: a sunyntax-directed Programming Environment , *Communications of the ACM*, vol. 24, n. 9, 1981, p. 152–168.

[THE 92] THERY L. AND BERTOT L. AND KAHN G., Real Theorem Provers Deserve Real User-Interfaces , DOOLEY S., *Proceedings of the Fifth ACM Symposium on Software Development Environments (SDE5)*, Washington D. C, dec 1992, p. 283–290.

[TOP 99] TOPPING P., Using MathType to create TeX and MathML equations , *TUGboat*, vol. 20, n. 3, 1999, p. 184–188.

[VAT 00] VATTON I., W3C's Amaya 4.0 Editor Browser , , 2000, W3C, http://w3c1.inria.fr/Amaya/.

[WEB] WebEQ , `http://www.webeq.com/`.

[YOU 87] YOUNG D. A.,WANG P. S., GI/S: a graphical user interface for symbolic computation systems , *Journal of Symbolic Computation*, vol. 4, n. 3, 1987, p. 365–380.

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - B.P. 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Lorraine : Technopôle de Nancy-Brabois - Campus scientifi que
615, rue du Jardin Botanique - B.P. 101 - 54602 Villers lès Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot St Martin (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, B.P. 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399